Polynomial-Time Algorithms for Contiguous Art Gallery and Related Problems

Ahmad Biniaz, Anil Maheshwari, Magnus Christian Ring Merrild, Joseph S.B. Mitchell, Saeed Odak, Valentin Polishchuk, Eliot W. Robson, Casper Moldrup Rysgaard, Jens Kristian Refsgaard Schou, Thomas Shermer, Jack Spalding-Jamieson, Rolf Svenning, Da Wei Zheng

SoCG June 2025

Combined work

Combined work

The Contiguous Art Gallery Problem is Solvable in Polynomial Time,

- Merrild, Rysgaard, Schou & Svenning

Combined work

The Contiguous Art Gallery Problem is Solvable in Polynomial Time,

- Merrild, Rysgaard, Schou & Svenning

Contiguous Boundary Guarding

- Biniaz, Maheshwari, Mitchell, Odak, Polishchuk & Shermer

Combined work

The Contiguous Art Gallery Problem is Solvable in Polynomial Time,

- Merrild, Rysgaard, Schou & Svenning

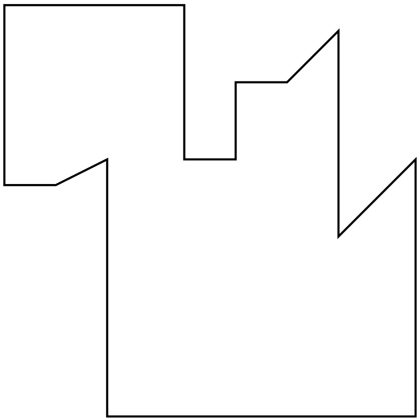
Contiguous Boundary Guarding

- Biniaz, Maheshwari, Mitchell, Odak, Polishchuk & Shermer

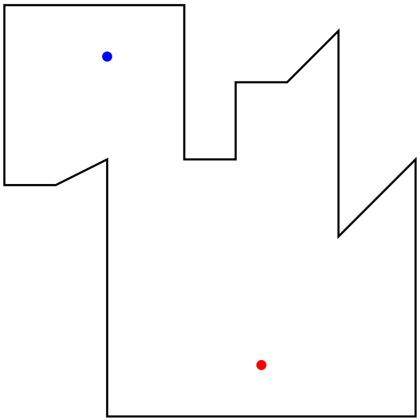
The Analytic Arc Cover Problem and its Applications to Contiguous Art Gallery, Polygon Separation, and Shape Carving

- Robson, Spalding-Jamieson & Zheng

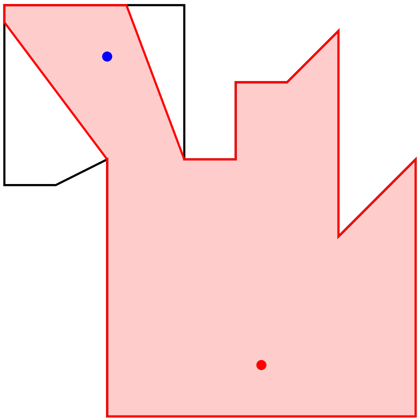
Art Gallery



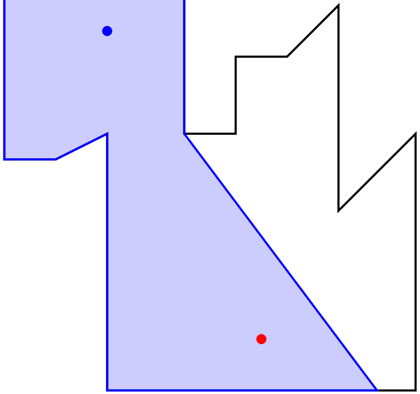
Art Gallery



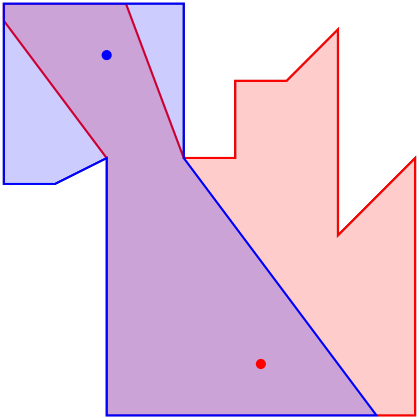
Art Gallery



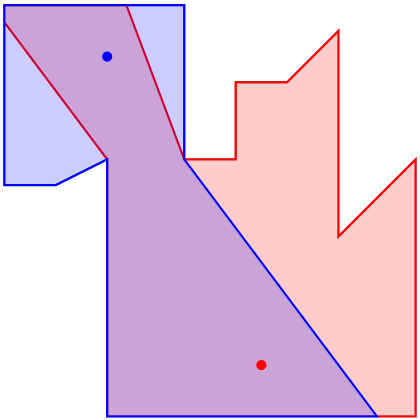
Art Gallery



Art Gallery



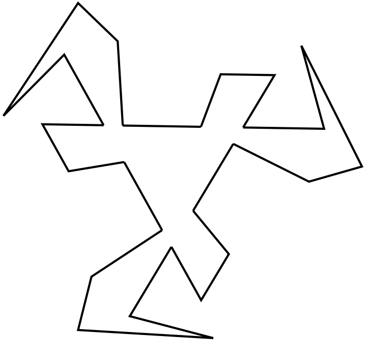
Art Gallery



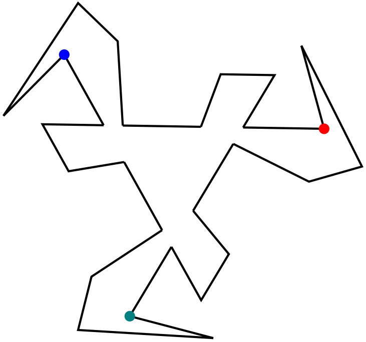
$\exists\mathbb{R}$-complete

[Abrahamsen, Adamaszek Miltzow, 2021]

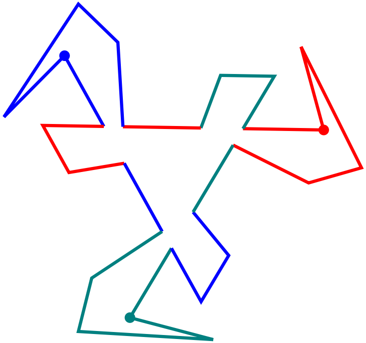
Boundary Guarding



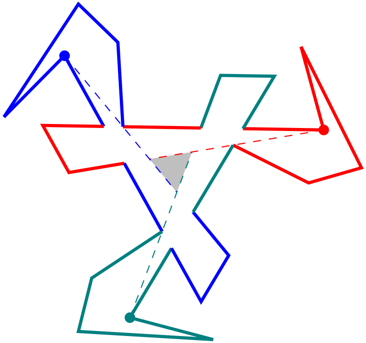
Boundary Guarding



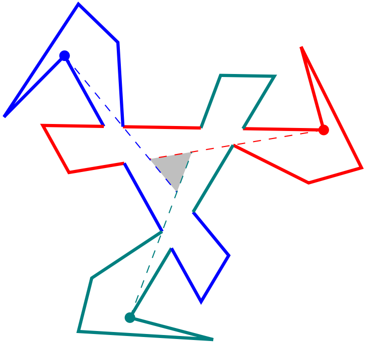
Boundary Guarding



Boundary Guarding



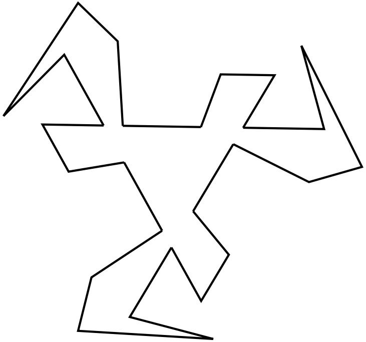
Boundary Guarding



$\exists\mathbb{R}$-complete

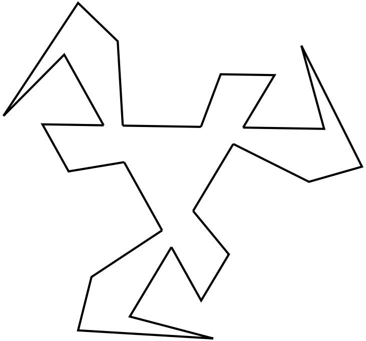
[Stade, SoCG25]

The Contiguous Art Gallery Problem



Goal: Guard boundary (few guards)

The Contiguous Art Gallery Problem

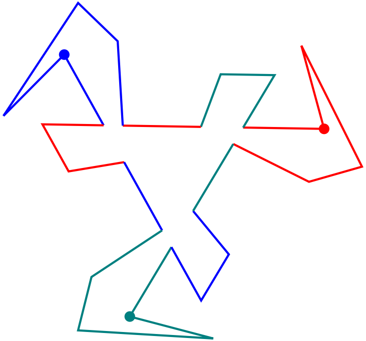


Goal: Guard boundary (few guards)

Restriction:

- Guards assigned to contiguous parts of boundary

The Contiguous Art Gallery Problem



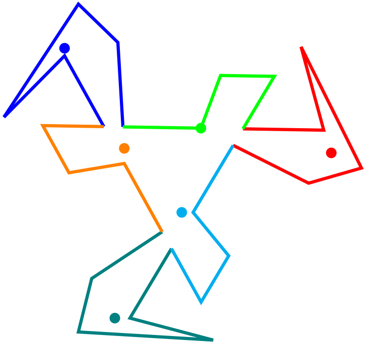
Goal: Guard boundary (few guards)

Restriction:

- Guards assigned to contiguous parts of boundary

(Not allowed)

The Contiguous Art Gallery Problem



Goal: Guard boundary (few guards)

Restriction:

- Guards assigned to contiguous parts of boundary

(Allowed)

Combinatorial Bounds

Combinatorial Bounds

Normal art gallery/boundary guarding: need $\leq \lfloor \frac{n}{3} \rfloor$ guards.

Combinatorial Bounds

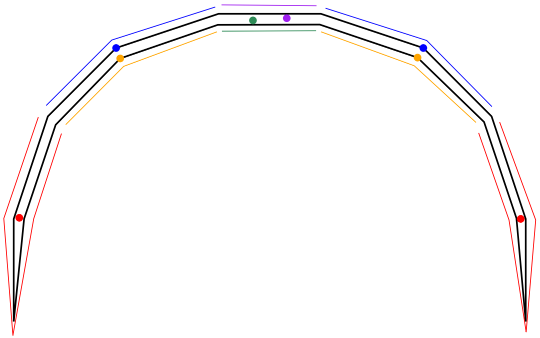
Normal art gallery/boundary guarding: need $\leq \lfloor \frac{n}{3} \rfloor$ guards.

Contiguous art gallery: need $\leq \lfloor \frac{n-2}{2} \rfloor$ guards.

Combinatorial Bounds

Normal art gallery/boundary guarding: need $\leq \lfloor \frac{n}{3} \rfloor$ guards.

Contiguous art gallery: need $\leq \lfloor \frac{n-2}{2} \rfloor$ guards.

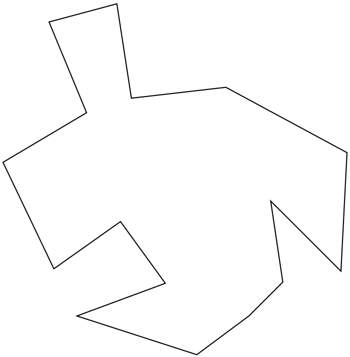


The Greedy Algorithm

“Take greedy steps”

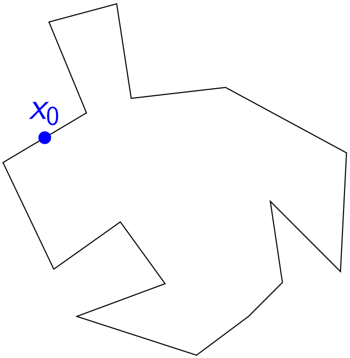
The Greedy Algorithm

"Take greedy steps"



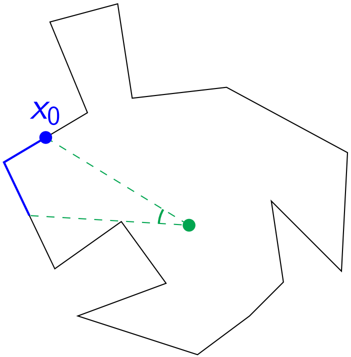
The Greedy Algorithm

"Take greedy steps"



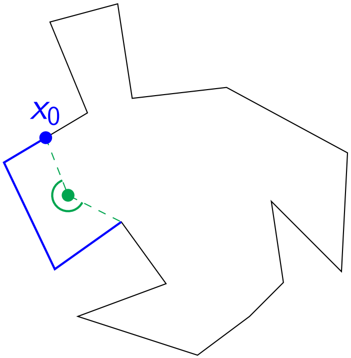
The Greedy Algorithm

"Take greedy steps"



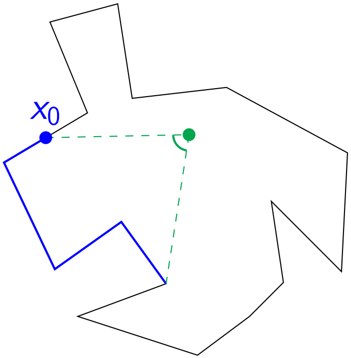
The Greedy Algorithm

"Take greedy steps"



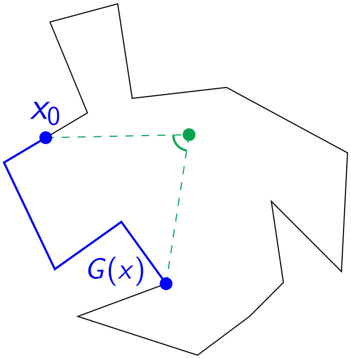
The Greedy Algorithm

"Take greedy steps"



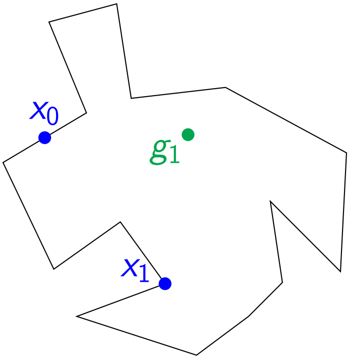
The Greedy Algorithm

"Take greedy steps"



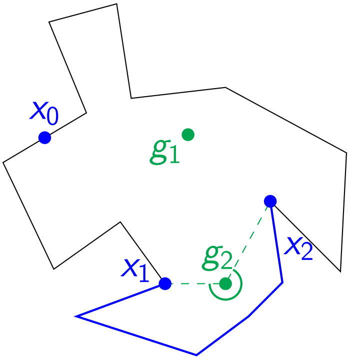
The Greedy Algorithm

"Take greedy steps"



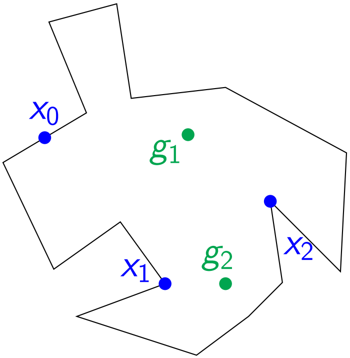
The Greedy Algorithm

"Take greedy steps"



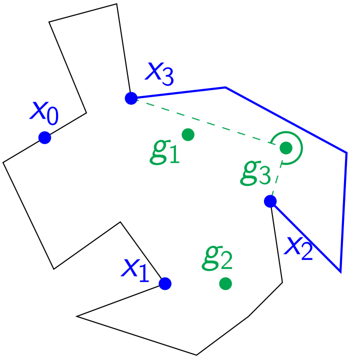
The Greedy Algorithm

"Take greedy steps"



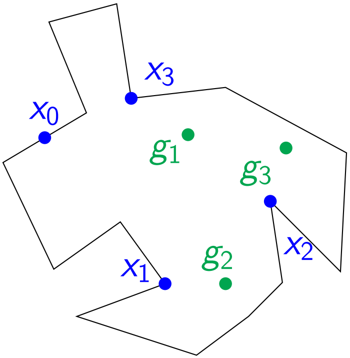
The Greedy Algorithm

"Take greedy steps"



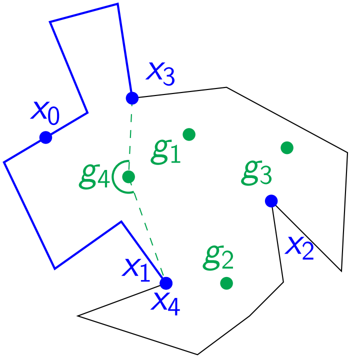
The Greedy Algorithm

"Take greedy steps"



The Greedy Algorithm

"Take greedy steps"



The Greedy Algorithm

One revolution: $\mathcal{O}\left(n^2 \log(n)\right)$ time.

The Greedy Algorithm

One revolution: $\mathcal{O}\left(n^2 \log(n)\right)$ time.

Guard count: **OPT** or **OPT+1**, depending on starting point.

The Greedy Algorithm

One revolution: $\mathcal{O}\left(n^2 \log(n)\right)$ time.

Guard count: **OPT** or **OPT+1**, depending on starting point.

Exact solution attainable?

Exact algorithms

Theorem (Main theorem)

Exact algorithms

Theorem (Main theorem)

The contiguous art gallery problem is solvable in polynomial time.

Exact algorithms

Theorem (Main theorem)

The contiguous art gallery problem is solvable in polynomial time.

Poly-time on TM for rational coordinates in input...

Exact algorithms

Theorem (Main theorem)

The contiguous art gallery problem is solvable in polynomial time.

Poly-time on TM for rational coordinates in input...

but stated time complexities will be for real RAM.

Exact algorithms

Theorem (Main theorem)

The contiguous art gallery problem is solvable in polynomial time.

Poly-time on TM for rational coordinates in input...

but stated time complexities will be for real RAM.

Three different algorithms/methods.

Overview of Solutions

Three different algorithms/methods

Common element: Greedy steps

Overview of Solutions

Three different algorithms/methods

Common element: Greedy steps

- Method 1: Perform more greedy steps

Overview of Solutions

Three different algorithms/methods

Common element: Greedy steps

- Method 1: Perform more greedy steps
- Method 2: Select good candidate starting points

Overview of Solutions

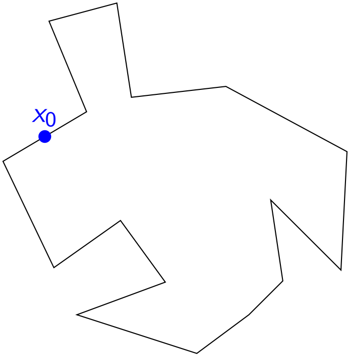
Three different algorithms/methods

Common element: Greedy steps

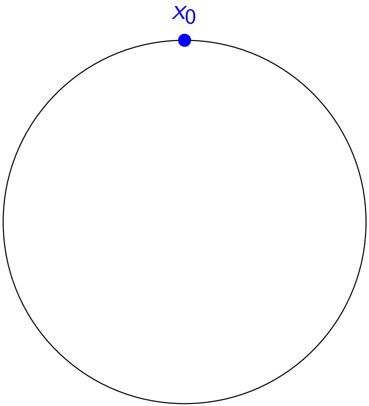
- Method 1: Perform more greedy steps
- Method 2: Select good candidate starting points
- Method 3: Try every starting point simultaneously

Method 1: Repeated Greedy

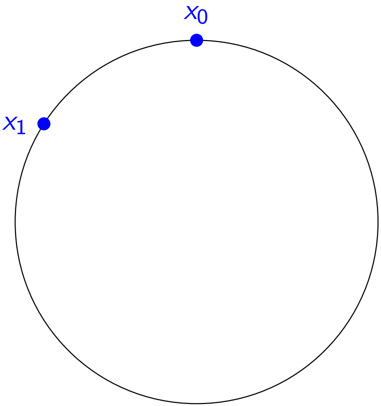
Method 1: Repeated Greedy



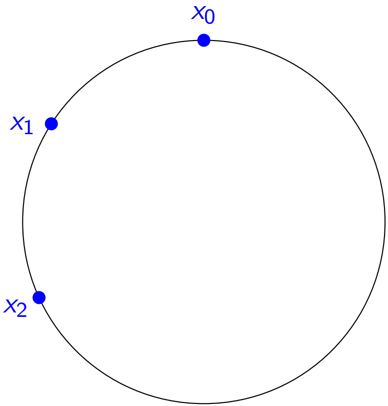
Method 1: Repeated Greedy



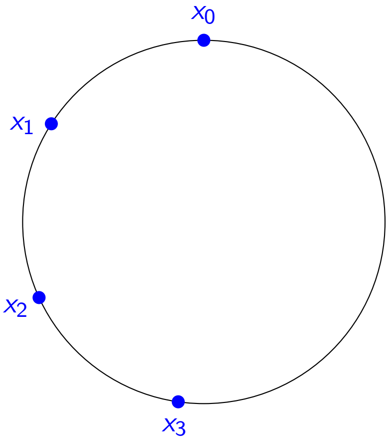
Method 1: Repeated Greedy



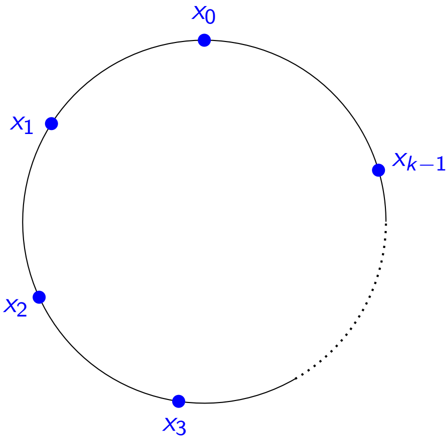
Method 1: Repeated Greedy



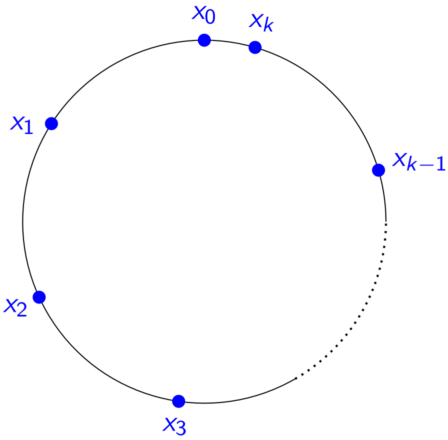
Method 1: Repeated Greedy



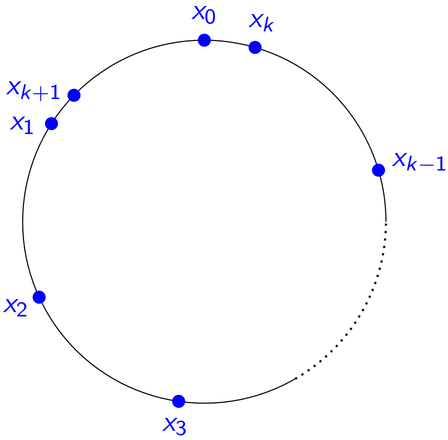
Method 1: Repeated Greedy



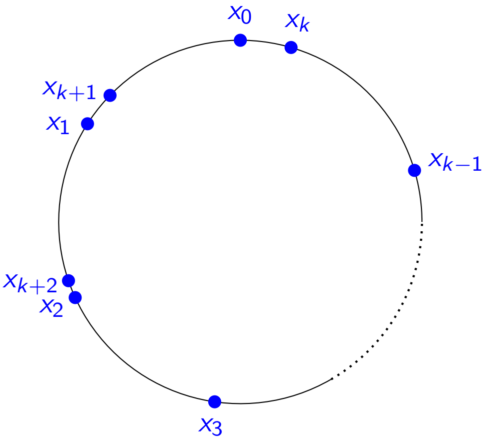
Method 1: Repeated Greedy



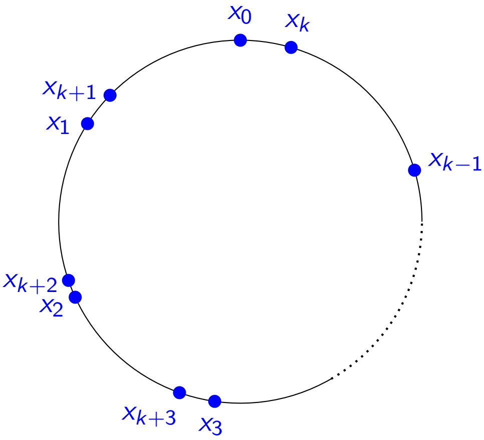
Method 1: Repeated Greedy



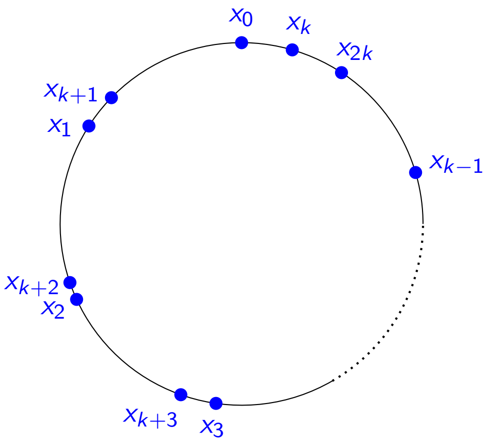
Method 1: Repeated Greedy



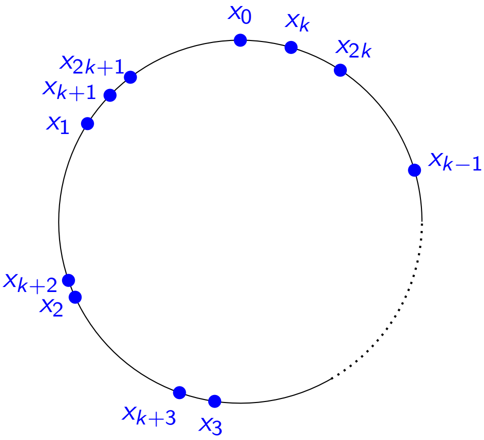
Method 1: Repeated Greedy



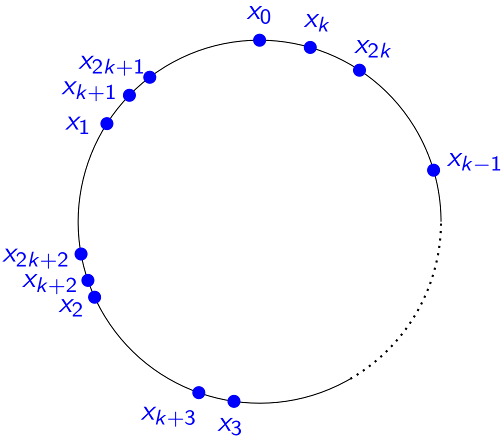
Method 1: Repeated Greedy



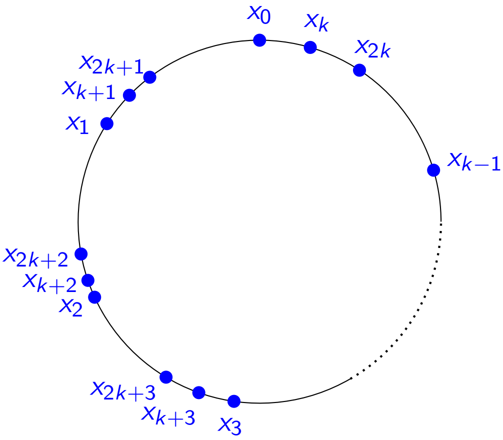
Method 1: Repeated Greedy



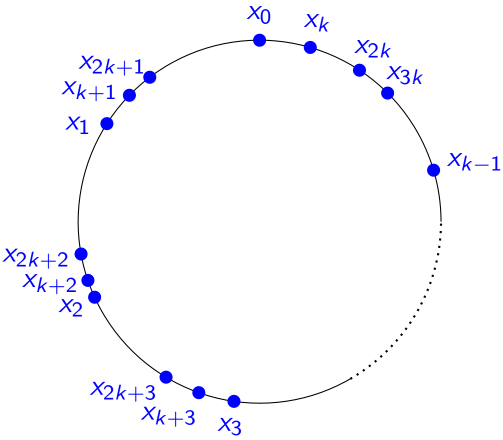
Method 1: Repeated Greedy



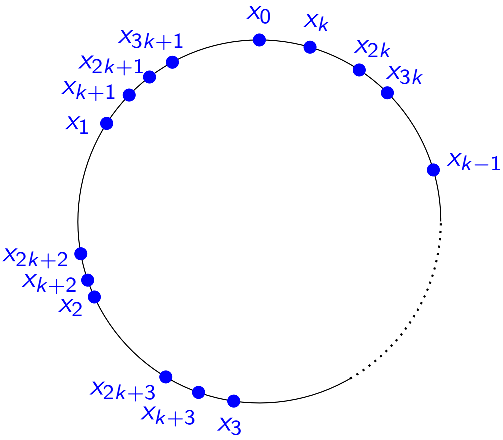
Method 1: Repeated Greedy



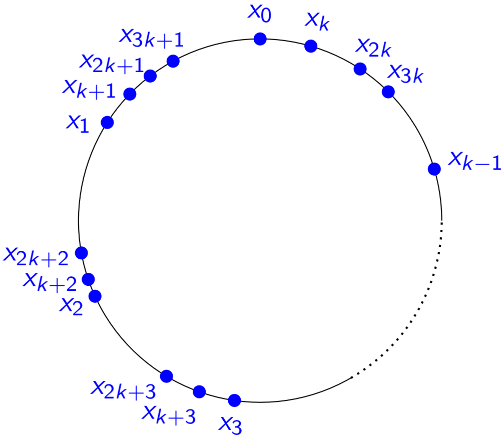
Method 1: Repeated Greedy



Method 1: Repeated Greedy

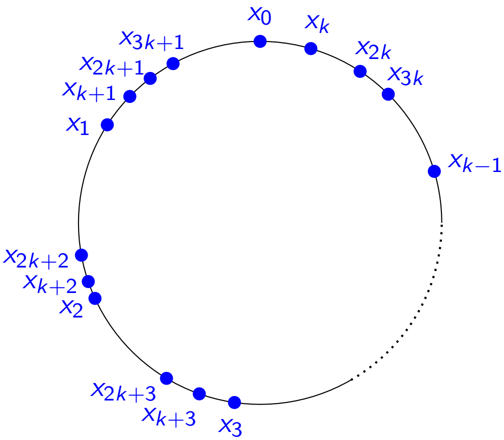


When to stop?



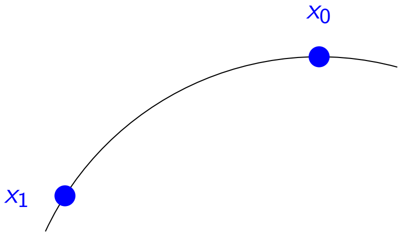
When to stop?

1. Repetition



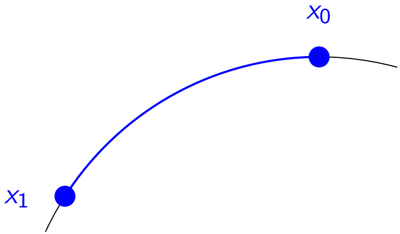
When to stop?

1. Repetition



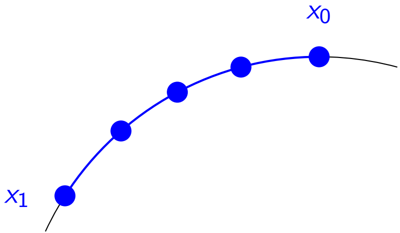
When to stop?

1. Repetition



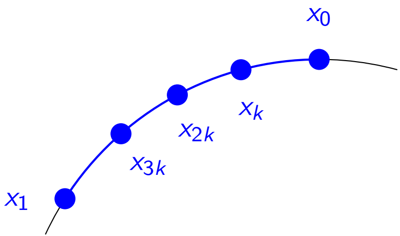
When to stop?

1. Repetition



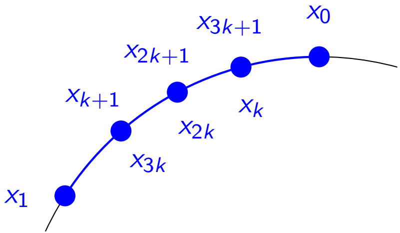
When to stop?

1. Repetition



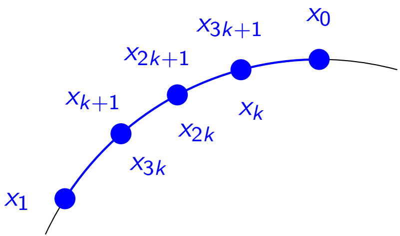
When to stop?

1. Repetition



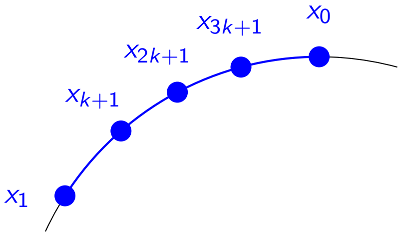
When to stop?

1. Repetition
2. $x_k, x_{2k}, x_{3k}, \cdots \to x_1$



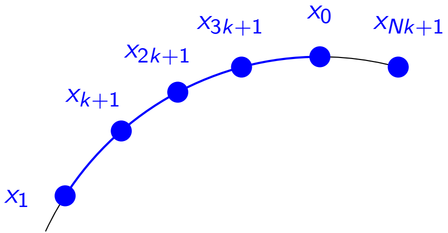
When to stop?

1. Repetition
2. $x_k, x_{2k}, x_{3k}, \cdots \to x_1$



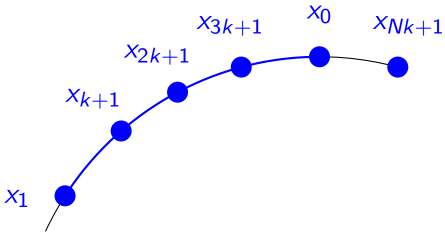
When to stop?

1. Repetition
2. $x_k, x_{2k}, x_{3k}, \cdots \to x_1$



When to stop?

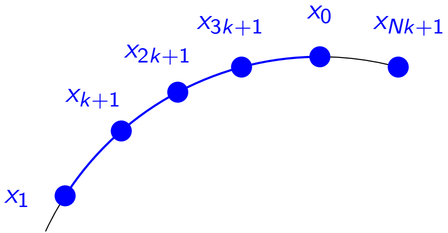
1. Repetition
2. $x_k, x_{2k}, x_{3k}, \cdots \to x_1$
3. $x_{k+1}, x_{2k+1}, x_{3k+1}, \ldots$ pass x_0



When to stop?

1. Repetition
2. $x_k, x_{2k}, x_{3k}, \cdots \to x_1$
3. $x_{k+1}, x_{2k+1}, x_{3k+1}, \ldots$ pass x_0

Guaranteed in $\mathcal{O}(n^3\mathbf{OPT})$ revolutions.

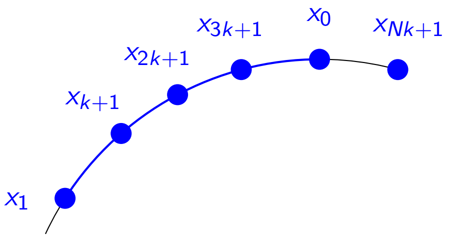


When to stop?

1. Repetition
2. $x_k, x_{2k}, x_{3k}, \cdots \to x_1$
3. $x_{k+1}, x_{2k+1}, x_{3k+1}, \ldots$ pass x_0

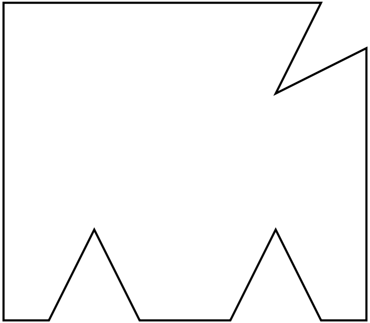
Guaranteed in $\mathcal{O}(n^3\mathbf{OPT})$ revolutions.

$\mathcal{O}(n^5\mathbf{OPT}\log n)$ runtime.

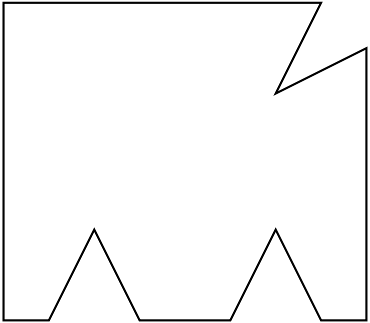


Method 2: Using one findable Guard

Method 2: Using one findable Guard

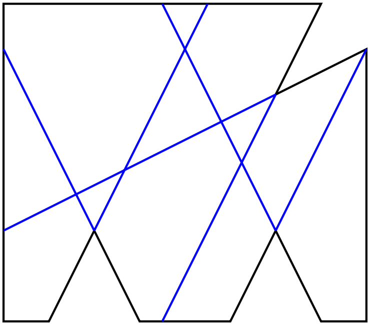


Method 2: Using one findable Guard



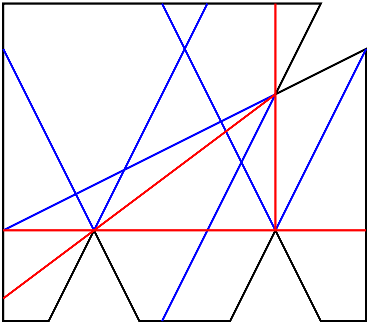
Shows existance of optimal solutions, with at least one guard at an incidence point.

Method 2: Using one findable Guard



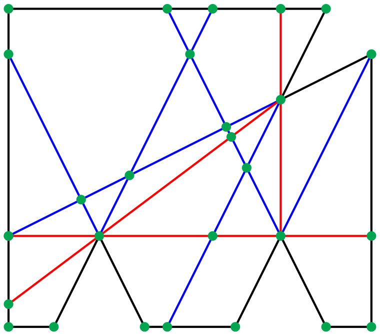
Shows existance of optimal solutions, with at least one guard at an incidence point.

Method 2: Using one findable Guard



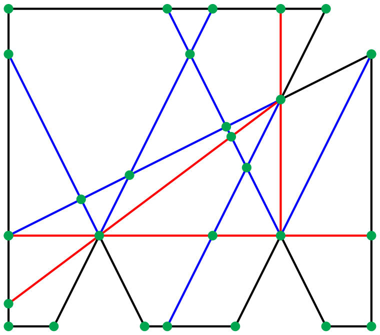
Shows existance of optimal solutions, with at least one guard at an incidence point.

Method 2: Using one findable Guard



Shows existance of optimal solutions, with at least one guard at an incidence point.

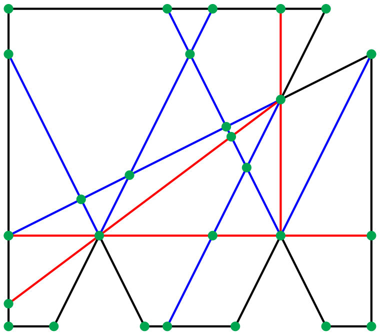
Method 2: Using one findable Guard



Shows existance of optimal solutions, with at least one guard at an incidence point.

At most $\mathcal{O}(n^3)$ incidence points.

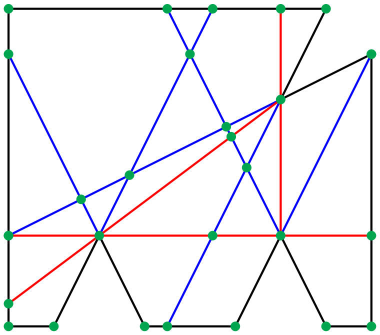
Method 2: Using one findable Guard



Shows existance of optimal solutions, with at least one guard at an incidence point.

At most $\mathcal{O}(n^3)$ incidence points. Try them all!

Method 2: Using one findable Guard

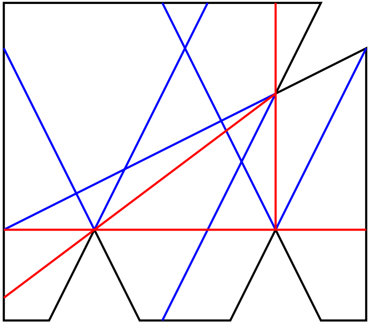


Shows existance of optimal solutions, with at least one guard at an incidence point.

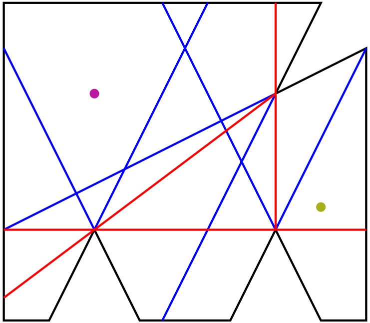
At most $\mathcal{O}(n^3)$ incidence points. Try them all!

Total running time: $\mathcal{O}(n^6 \log(n))$.

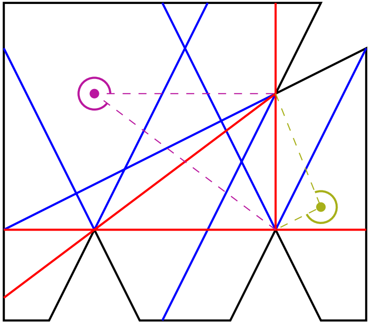
Sketch of Proof



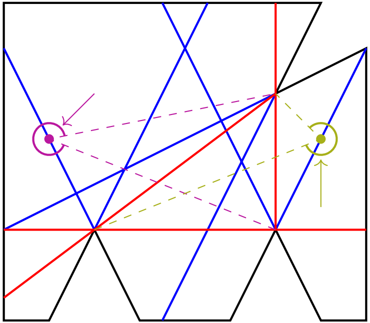
Sketch of Proof



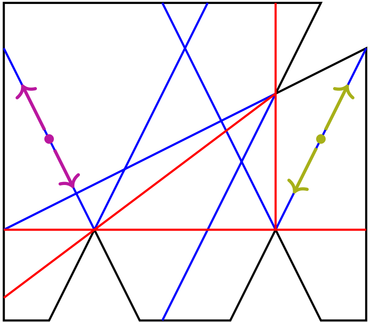
Sketch of Proof



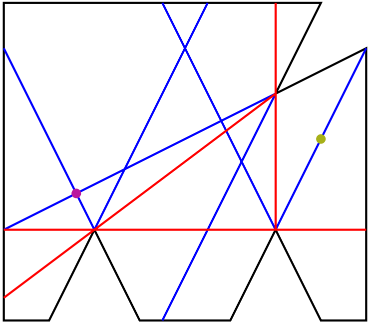
Sketch of Proof



Sketch of Proof



Sketch of Proof

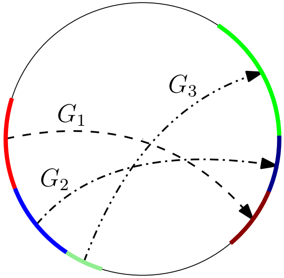


Method 3: Closed-Form Analysis ("Analytic Arc Cover Framework")

Idea: Find a closed form expression of greedy step G to evaluate *every* starting point simultaneously.

Method 3: Closed-Form Analysis ("Analytic Arc Cover Framework")

Idea: Find a closed form expression of greedy step G to evaluate *every* starting point simultaneously.



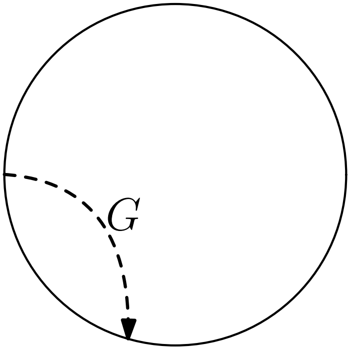
$$G_1 = \frac{ax+b}{cx+d},\ G_2 = \frac{a'x+b'}{c'x+d'},\ G_3 = \frac{a''x+b''}{c''x+d''}, \ldots$$

Bounds: $[a_0 + b_0\sqrt{c_0},\ a_1 + b_1\sqrt{c_1}),\ [a_1 + b_1\sqrt{c_1}, a_2 + b_2\sqrt{c_2}),\ [a_2 + b_2\sqrt{c_2}, a_3 + b_3\sqrt{c_3}), \ldots$

Using a closed form representation ("Analytic Arc Cover Framework")

Input: Closed form expression for G

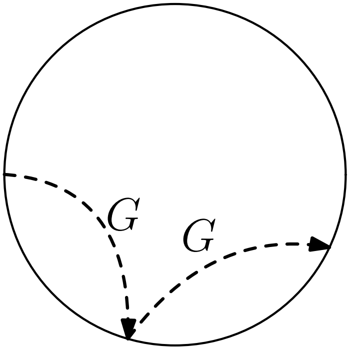
Algorithm: Repeatedly compose G with itself...



Using a closed form representation (“Analytic Arc Cover Framework”)

Input: Closed form expression for G

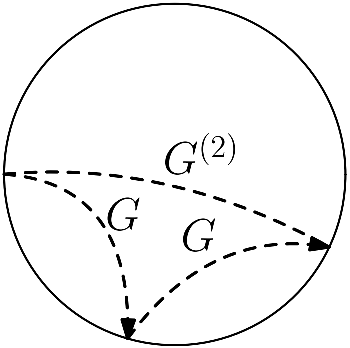
Algorithm: Repeatedly compose G with itself…



Using a closed form representation ("Analytic Arc Cover Framework")

Input: Closed form expression for G

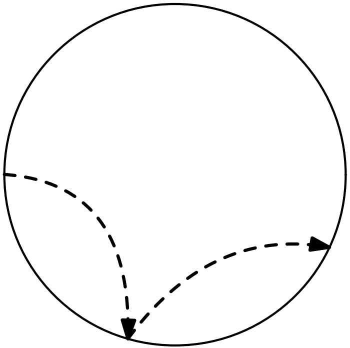
Algorithm: Repeatedly compose G with itself...



Using a closed form representation ("Analytic Arc Cover Framework")

Input: Closed form expression for G

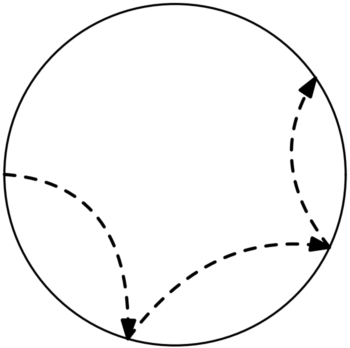
Algorithm: Repeatedly compose G with itself…



Using a closed form representation ("Analytic Arc Cover Framework")

Input: Closed form expression for G

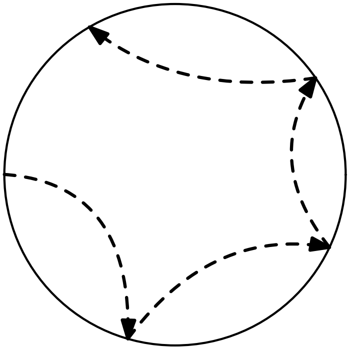
Algorithm: Repeatedly compose G with itself. . .



Using a closed form representation (“Analytic Arc Cover Framework”)

Input: Closed form expression for G

Algorithm: Repeatedly compose G with itself...

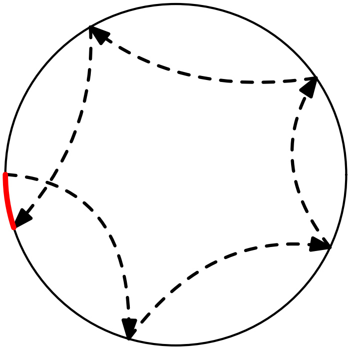


Using a closed form representation ("Analytic Arc Cover Framework")

Input: Closed form expression for G

Algorithm: Repeatedly compose G with itself...
until $G^{(k)}$ exceeds a full loop somewhere.

Output: k



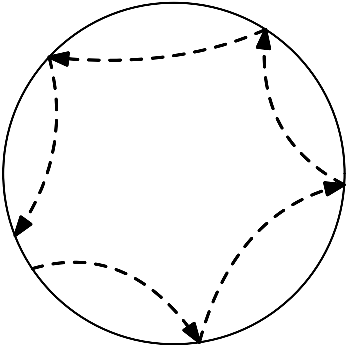
Using a closed form representation ("Analytic Arc Cover Framework")

Input: Closed form expression for G

Algorithm: Repeatedly compose G with itself...

until $G^{(k)}$ exceeds a full loop somewhere.

Output: k



Need to test $G^{(k)}$ everywhere at once.

Analytic Arc Cover Framework: Benefits

- Only need to find one function

Analytic Arc Cover Framework: Benefits

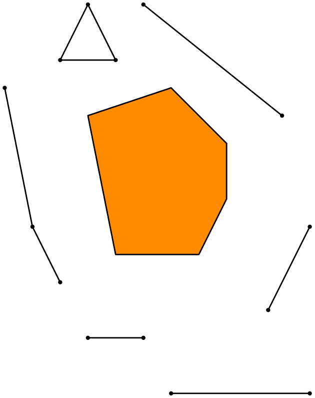
- Only need to find one function
- Robust to some problem variants

Analytic Arc Cover Framework: Benefits

- Only need to find one function
- Robust to some problem variants
- Also applicable to some other problems. . .

More problems: Line Segment/Polygon Separation

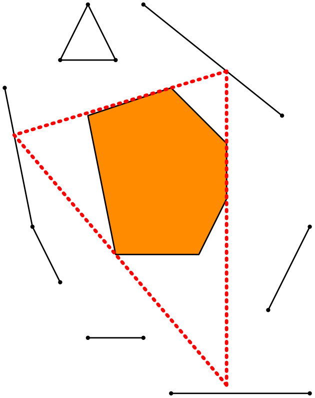
Input: A convex polygon and a set of line segments.



More problems: Line Segment/Polygon Separation

Input: A convex polygon and a set of line segments.

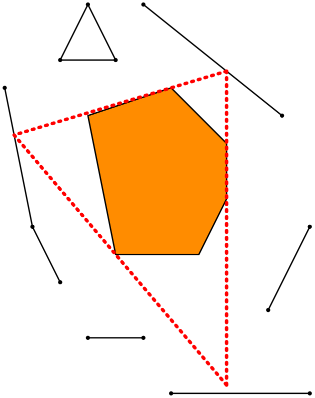
Output: A convex polygon separating the two, minimizing vertices.



More problems: Line Segment/Polygon Separation

Input: A convex polygon and a set of line segments.

Output: A convex polygon separating the two, minimizing vertices.



Solvable with analytic arc cover framework!

More Problems: Min Half-Plane Carving

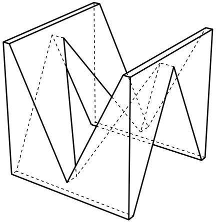
- What shapes can be carved from a large block of wood with half-plane cuts? (solved at CCCG '24)

More Problems: Min Half-Plane Carving

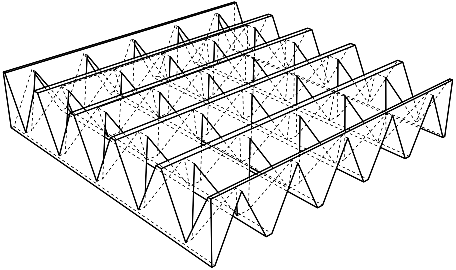
- What shapes can be carved from a large block of wood with half-plane cuts? (solved at CCCG '24)
- How many cuts are needed? (new)

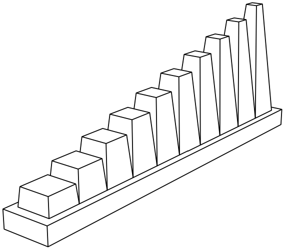
More Problems: Min Half-Plane Carving

- What shapes can be carved from a large block of wood with half-plane cuts? (solved at CCCG '24)
- How many cuts are needed? (new)



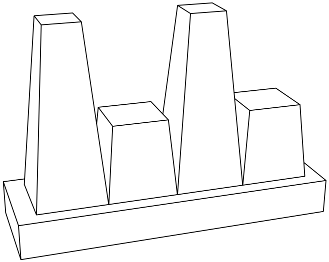
Carveable:

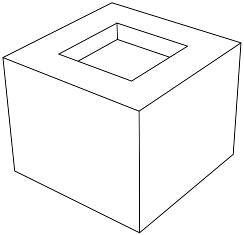




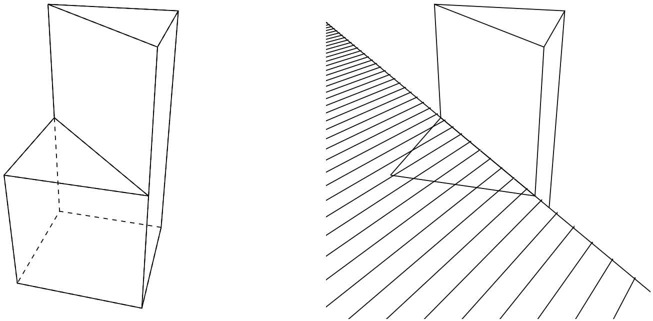
More Problems: Min Half-Plane Carving

Uncarveable:



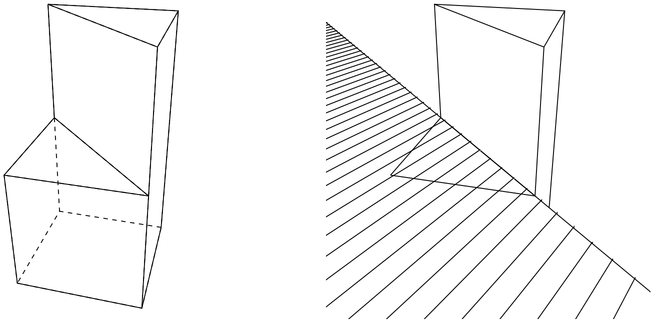


Min Half-Plane Cutting: Reducing to 2D



- Minimizing in 3D $\to$ Minimizing (many) in 2D

Min Half-Plane Cutting: Reducing to 2D



- Minimizing in 3D $\rightarrow$ Minimizing (many) in 2D
- Reduction to line segment/polygon separation

Future Work

Future Work

Faster runtime:

Future Work

Faster runtime:

- Almost linear runtime for greedy revolution

Future Work

Faster runtime:

- Almost linear runtime for greedy revolution
- Way fewer revolutions for Method 1

Future Work

Faster runtime:

- Almost linear runtime for greedy revolution
- Way fewer revolutions for Method 1

Other variants:

Future Work

Faster runtime:

- Almost linear runtime for greedy revolution
- Way fewer revolutions for Method 1

Other variants:

- Higher dimensional galleries.

Future Work

Faster runtime:

- Almost linear runtime for greedy revolution
- Way fewer revolutions for Method 1

Other variants:

- Higher dimensional galleries.
- Each guard guards at most m polygonal arcs.

Thank you [1]

[1] Also Jack is looking for PhD positions. . .